

USBprog 5.0

Stand 18.12.2014 / Dokument Version V108

© embedded projects GmbH

<http://www.embedded-projects.net>

Inhaltsverzeichnis

1. Historie.....	4
2. Überblick USBprog 5.0.....	4
2.1. Hardware Überblick.....	4
2.2. Prozessor Adapterplatinen.....	5
3. Quickstart.....	6
4. Installation.....	7
4.1. Linux.....	7
4.2. Mac.....	7
4.2.1. Systemvoraussetzungen.....	7
4.2.2. Installation des USB-Netzwerks.....	7
4.2.3. Installation von Python.....	7
4.3. Windows 7.....	7
4.3.1. Installation des USB-Netzwerks.....	7
4.3.2. Installation von embeddedprog.exe.....	11
4.4. Andere Betriebssysteme.....	11
5. Bedienung.....	12
5.1. Bedienung per Browser.....	12
5.1.1. Allgemeines.....	12
5.1.2. Wählen der Settings.....	12
5.1.3. Benutzung des „Commands“-Bereich.....	13
5.1.4. Programmieren einer Firmware.....	13
5.1.5. Nutzung des Flash Archivs.....	14
5.1.6. Einspielen eines Updates für USBprog 5.0.....	14
5.1.7. Kommandozeilen Befehle.....	14
5.2. Bedienung Per Konsole.....	15
5.2.1. Öffnen der Konsole.....	15
5.2.2. Download des Kommandozeilentools.....	16
5.2.3. Verwendung des Kommandozeilentools.....	17
5.2.4. Übersicht der Befehle.....	17
6. Einrichten des USBprog 5.0 für AtmelStudio.....	19
6.1. Einrichten der Funktionen.....	19
6.1.1. Funktion Program.....	20
6.1.2. Funktion Open Settings.....	21
6.2. Benutzung der Funktionen.....	22
7. Update Firmware.....	24
8. Debuggen Mit OpenOCD.....	24
8.1. Debuggen mit Kommandozeile.....	24
8.1.1. Linux.....	25
8.2. Debuggen mit Eclipse.....	28
9. USBprog 5.0 Systemdetails.....	28
9.1. Open-Source Programme.....	28
9.2. Buchsen GPIO Belegung.....	28
10. API.....	29
10.1. JSON Befehle.....	29
10.2. JSON Beispiel.....	29
10.2.1. Signatur abfragen.....	29
10.2.2. Firmware flashen.....	30

1. Historie

Datum	Version	Bemerkung	Bearbeiter
17.12.2014	V107	Ergänzung ARM debuggen per Konsole	Jannis Barth
18.12.2014	V108	Diverse Anpassungen im Text	Benedikt Sauter

2. Überblick USBprog 5.0

Der USBprog 5.0 OpenOCD ist ein integrierter Programmierer in dem sowohl Hardware als auch Software enthalten ist. Durch die USB Verbindung zum PC wird ein Netzwerk aufgebaut über das man den Programmierer einfach bedienen kann. Die Bedienung kann über ein Kommandozeilentool, die API oder direkt über den Browser erfolgen. Mit diesem Programmierer ist es möglich sowohl AVR als auch ARM Prozessoren zu programmieren. Außerdem ist es einfach möglich über den Programmierer den integrierten OpenOCD Debugger zu starten und damit zu Debuggen.

Zusätzliche Features wie das Flash-Archiv ermöglichen es einfach eine feste Version einer Firmware über die Weboberfläche einzuspielen. Das Flash-Archiv wird auf dem USBprog 5.0 gespeichert.

2.1. Hardware Überblick

Auf dem USB Programmierer befinden sich drei Anschlüsse: USB (für Netzwerkemulation und Stromversorgung), 10 polige Programmier und Debugschnittstelle (Belegung siehe Abschnitt 10.2) und der 14-polige Standard GnuBlin/EEC Stecker für den es viele Erweiterungen gibt (z.B. Display, Motor, AD-Wandler, Temperaturfühler usw.).

Für die Anpassung der Spannungen an die verschiedenen Prozessoren gibt es einen einstellbaren Spannungsregler. Über die Oberflächen können Standardspannungen von 1.8V, 3.3V und 5.0V erzeugt werden.

Auf dem restlichen Teil der Schaltung befindet sich ein Prozessor und Arbeitsspeicher auf dem das Linux mit der Software läuft. Das Linux wird von der SD-Karte gebootet. Updates spielt man über die Weboberfläche ein. Regelmäßig werden neue Updates im GIT online gestellt. Sollte doch mal etwas schief gehen kann man auf das komplette Image aus unserem Download Bereich zugreifen.

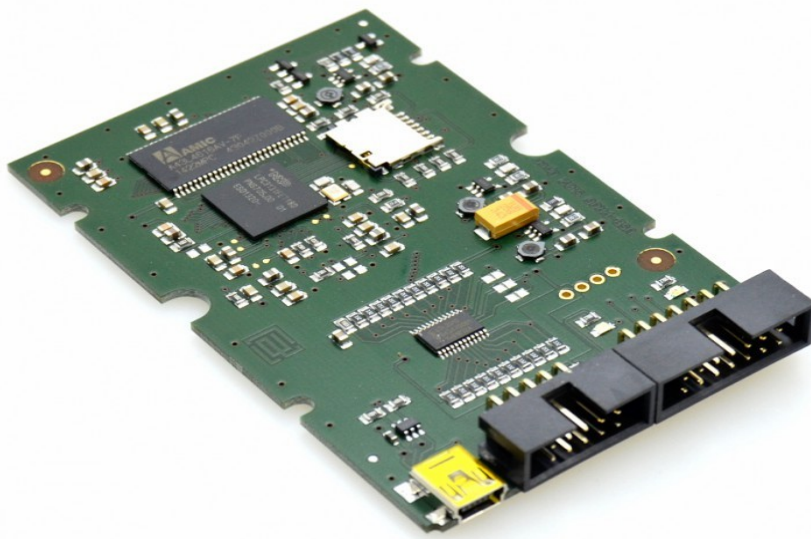


Abbildung 1: USBprog 5.0

2.2. Prozessor Adapterplatinen

Die Prozessoren werden über den 10 poligen Programmier- und Debugstecker angebunden. Die Belegung ist die Standardbelegung aller USBprog Versionen. Sie entspricht der 10 poligen AVR ISP Buchse. Das bedeutet, liegt in der zu programmierenden Schaltung ein 10 poliger Anschluss für einen AVR Controller vor, kann dieser 1:1 verbunden werden.

Um auf 6 polige AVR Buchsen zugreifen zu können muss der Adapter 10 auf 6 (Abbildung 3) gewählt werden. Für ARM Prozessoren mit einem 20 poligen Anschluss muss der Adapter 10 zu 20 (Abbildung 2) gewählt werden. Es werden weitere Adapter wie z.B für PIC Prozessoren folgen.

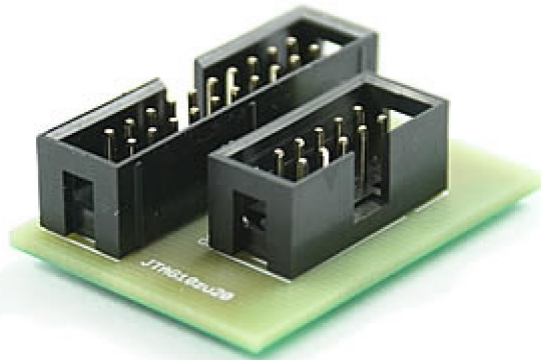


Abbildung 2: ARM 10 auf 20

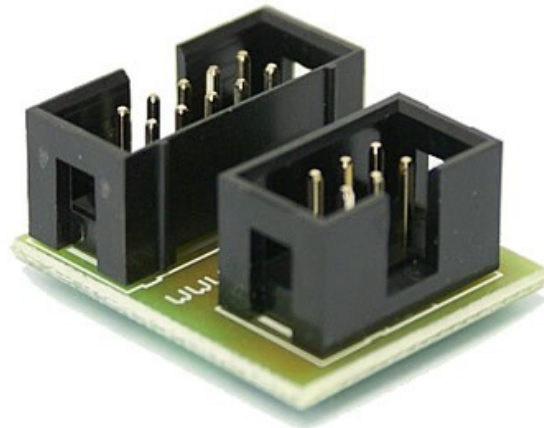


Abbildung 3: AVR 10 auf 6

3. Quickstart

Im USBprog befindet sich die komplette Hard- und Software die für das Programmieren von Mikroprozessoren benötigt wird. Die Funktionen erreicht man entweder per Browser oder dem Kommandozeilentool, bzw. für Experten auch per API oder SSH.

Die Prinzipielle vorgehensweise ist:

1. USBprog 5.0 per USB Kabel anstecken
2. Es wird, nachdem die Treiber installiert sind (siehe Kapitel 4) automatisch eine Netzwerkverbindung zwischen USBprog und PC aufgebaut
3. Die Rote LED geht sobald der USBprog betriebsbereit ist an
4. Verbinden per Browser: <http://10.0.0.1> oder mit dem Kommandozeilentool (Eventuell dauert es nochmal einige Sekunden bis das Betriebssystem die Netzwerkverbindung eingerichtet hat. Daher hier notfalls etwas Geduld.)

4. Installation

Um auf den USBprog 5.0 zuzugreifen muss ein Netzwerk über USB hergestellt werden. Falls das Kommandozeilentool benutzt werden soll, muss dieses vom GIT Server oder der lokalen Webseite des USBprog 5.0 OpenOCD heruntergeladen werden.

Standardmäßig ist die IP des USBprog 5.0 OpenOCD: 10.0.0.1

4.1. Linux

Durch Anstecken des USBprog 5.0 über den USB-Port wird das neue USB-Netzwerk erkannt und man erhält, vom auf dem USBprog 5.0 laufendem DHCP-Server, automatisch eine IP.

Sollte dies nicht der Fall sein, ist es möglich, in der Konsole (STRG + ALT + t) den Network-Manager "von Hand" zu stoppen und sich selbst die entsprechende IP zuzuweisen.

Benötigte Kommandos:

```
sudo stop network-manager
sudo dhclient eth0
sudo ifconfig usb0 10.0.0.2 up
```

Sollte Python nicht installiert sein, öffnen Sie die Konsole (STRG + ALT + t) und geben sie dort folgendes ein:

```
sudo apt-get install python
```

Nach dem Anstecken dauert es ca. 10-30 Sekunden bis die rote LED auf dem USBprog 5.0 angeht. Jetzt kann man sich mit dem Browser oder Kommandozeilentool mit dem Programmer verbinden.

4.2. Mac

4.2.1. Systemvoraussetzungen

Ab MacOS X 10.6.3 und größer.

4.2.2. Installation des USB-Netzwerks

Der Adapter wird automatisch erkannt und mit einer IP-Adresse konfiguriert.

Nach dem Anstecken dauert es ca. 10-30 Sekunden bis die rote LED auf dem USBprog 5.0 angeht. Jetzt kann man sich mit dem Browser oder Kommandozeilentool mit dem Programmer verbinden.

4.2.3. Installation von Python

Python ist in den aktuelle MacOS X Versionen bereits vorhanden.

4.3. Windows 7

4.3.1. Installation des USB-Netzwerks

1. Klicken Sie auf den Windows Start Button und wählen dort "Systemsteuerung" aus (siehe Abbildung 4)

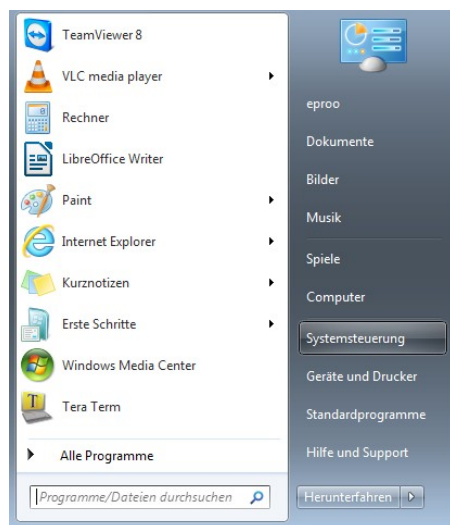


Abbildung 4: Systemsteuerung
öffnen

2. "Geräte Manager" auswählen. Falls Sie die Kategorie "Ansicht" verwenden ist der Geräte-Manager unter "Hardware und Sound" zu finden (siehe Abbildung 5)

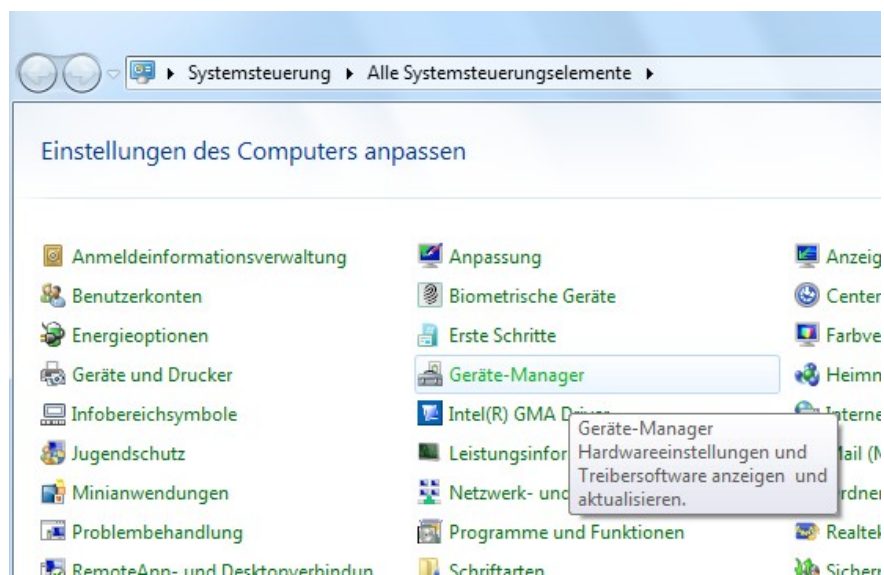


Abbildung 5: Geräte-Manager wählen

3. Rechtsklick auf den den "RNDIS / Ethernet Gadget" ausführen, der unter "Andere Geräte" liegt und auf "Treibersoftware aktualisieren" klicken.(siehe Abbildung 6)

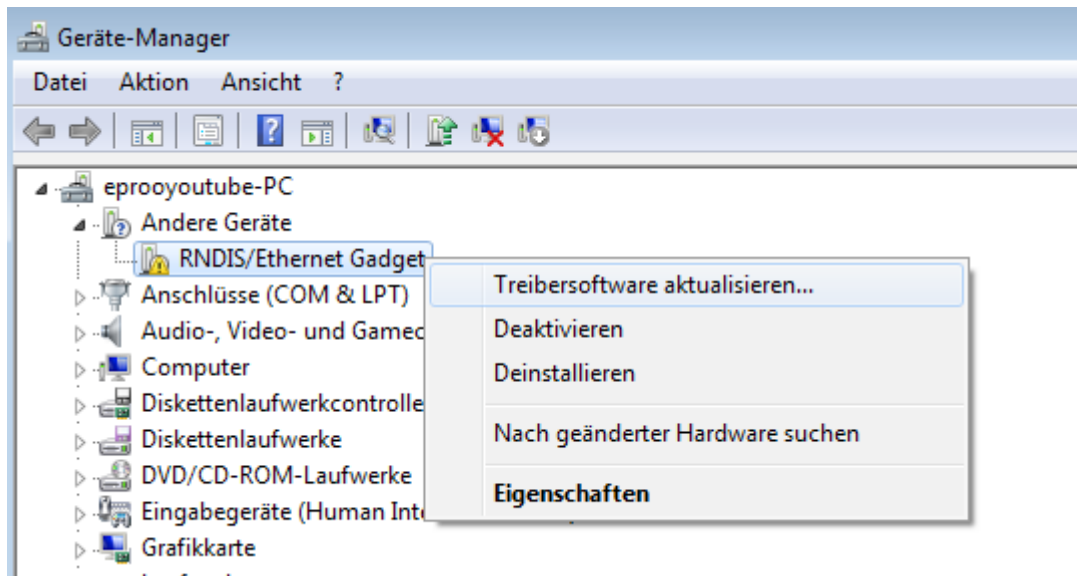


Abbildung 6: RNDIS Gadget aktualisieren

4. Klicken Sie auf: "Auf dem Computer nach Treibersoftware suchen" (siehe Abbildung 7)

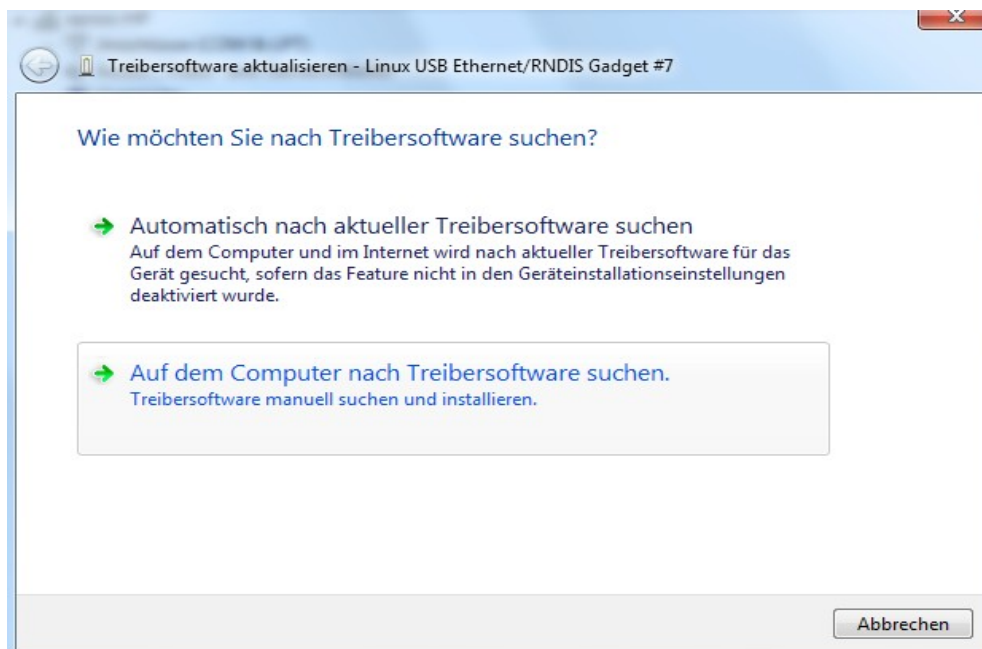


Abbildung 7: Auf dem Computer nach Software suchen

5. In dem neuen Fenster steht zur Auswahl, einen Pfad anzugeben, der aus einer Liste auswählbar ist. Wählen Sie hier aus einer Liste aus. (siehe Abbildung 8)

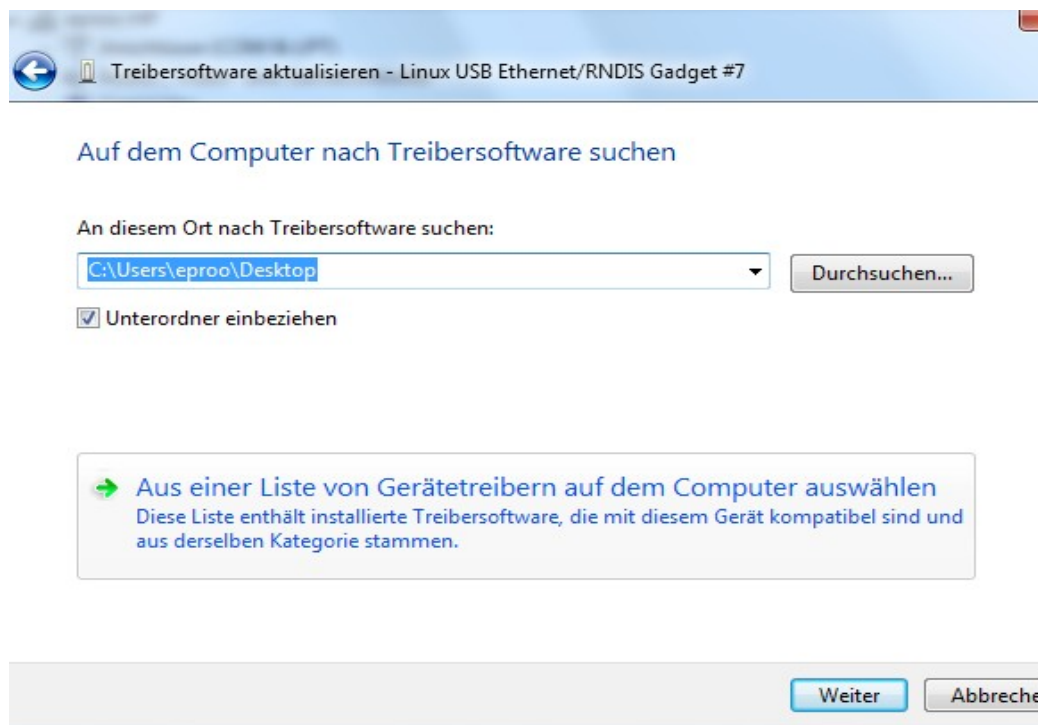


Abbildung 8: Aus Liste auswählen

6. Ein neues Fenster wird erscheinen in dem Sie den Typen des Geräts wählen sollen. Wählen Sie hier "Netzwerkadapter" (siehe Abbildung 9)

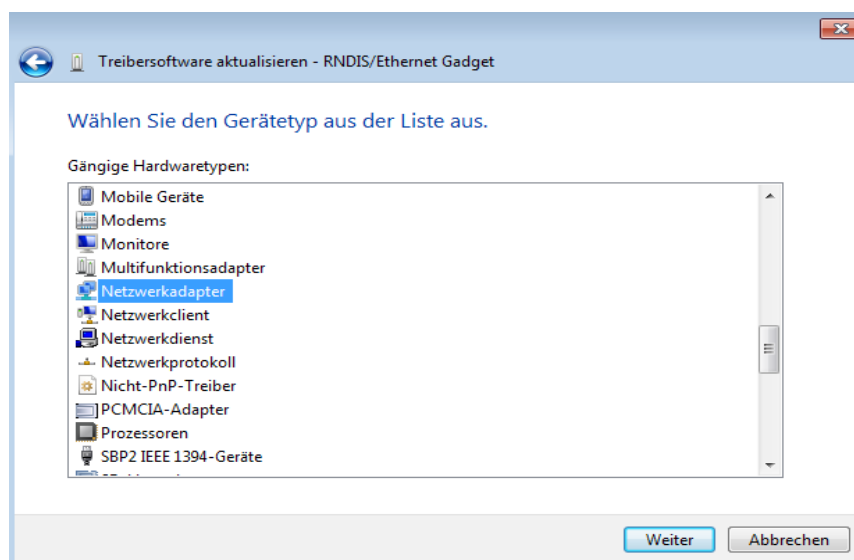


Abbildung 9: Netzwerkadapter

7. Im Folgenden Auswahlfenster wählen Sie "Microsoft Corporation" und im rechten Fenster "Remote NDIS based Internet Sharing Device". (siehe Abbildung 10)

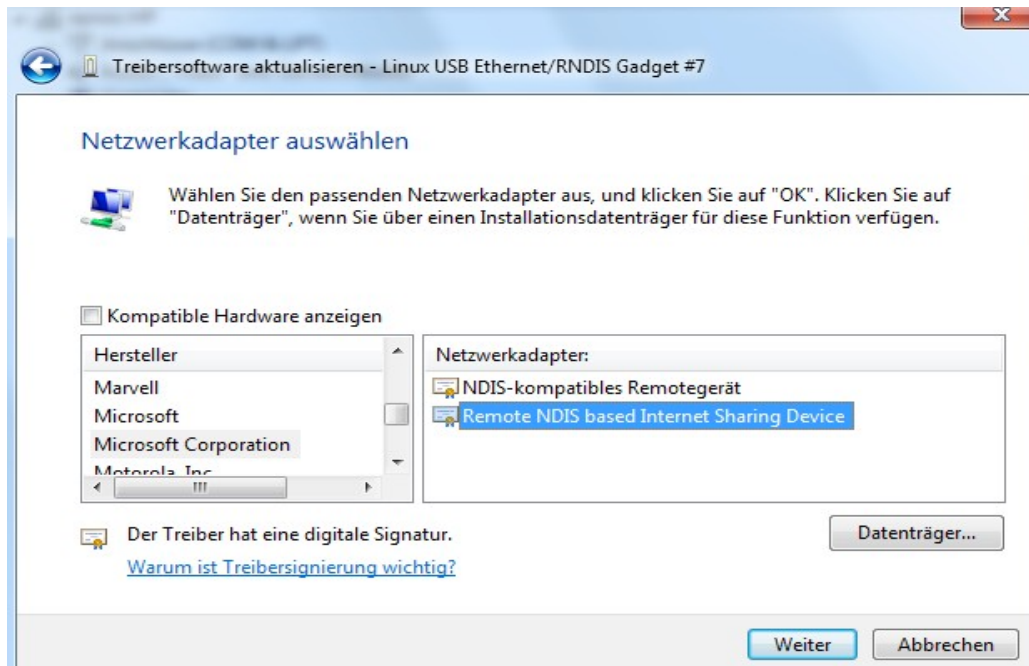


Abbildung 10: Microsoft Corporation Remote NDIS

8. Der USBprog 5.0 OpenOCD ist nun einsatzbereit.

Nach dem Anstecken dauert es ca. 10-30 Sekunden bis die rote LED auf dem USBprog 5.0 angeht. Jetzt kann man sich mit dem Browser oder Kommandozeilentool mit dem Programmer verbinden.

4.3.2. Installation von embeddedprog.exe

Die Datei „setup.exe“ kann von der internen Weboberfläche heruntergeladen werden (<http://10.0.0.1>).

4.4. Andere Betriebssysteme

Es wird mindestens Python 2.7 benötigt. Das Python muss die "argparse"-Funktion ausführen können. Das Betriebssystem muss dazu in der Lage sein ein USB-Netzwerk aufzubauen.

Nach dem Anstecken dauert es ca. 10-30 Sekunden bis die rote LED auf dem USBprog 5.0 angeht. Jetzt kann man sich mit dem Browser oder Kommandozeilentool mit dem Programmer verbinden.

5. Bedienung

Die Bedienung des USBprog 5.0 ist auf drei Arten möglich. Entweder über den Browser (siehe Abbildung 11), das Kommandozeilentool (siehe Abbildung 12) oder die API.

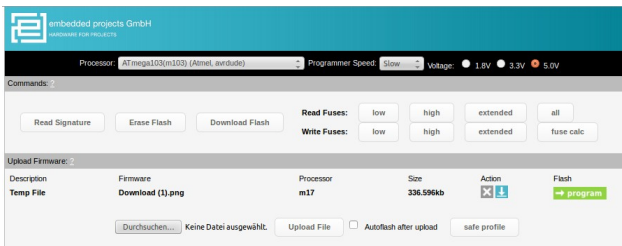


Abbildung 11: Browser

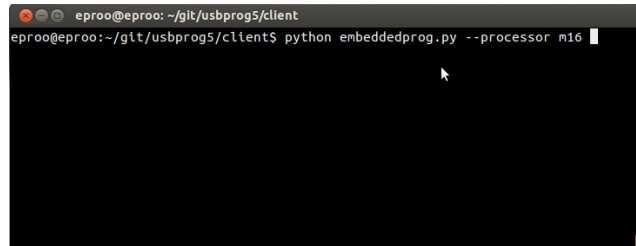


Abbildung 12: Konsole

5.1. Bedienung per Browser

Die Benutzung des USBprog 5.0 über den Browser ist am benutzerfreundlichsten, da diese Methode eine grafische Oberfläche bereitstellt.

5.1.1. Allgemeines

Standardmäßig ist die IP-Adresse des USBprog 5.0 OpenOCD "10.0.0.1" und ist über diese IP per Browser erreichbar.

5.1.2. Wählen der Settings

Im Settingsbereich (Abbildung 13) der Webseite können der Prozessor, die Spannung und die Programmier-Geschwindigkeit gewählt werden. Durch Klicken auf das Prozessorfeld kann nach einem Prozessor gesucht werden, in dem man den Namen des Prozessors eintippt oder den Prozessor im Drop-Down-Menü sucht. Das Drop-Down-Menü ist dynamisch, wodurch ihnen beim Eintippen alle Einträge, in denen das Eingetippte vorhanden ist, angezeigt werden.

Der „Programmer Speed“ ist momentan nur für AVR einstellbar und kann bei dem Programmieren eines ARM Prozessors vernachlässigt werden. Standardmäßig ist der „Programmer Speed“ auf "Normal" gesetzt.

Unter "Voltage" können Sie die benötigte Spannung auswählen.

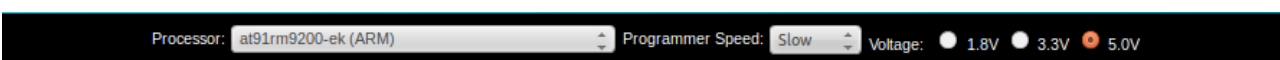


Abbildung 13: Settings

5.1.3. Benutzung des „Commands“-Bereich

Nachdem man einen Prozessor ausgewählt hat, wird der "Commands"-Bereich eingeblendet. Die Kommandos werden stets mit den ausgewählten Settings ausgeführt. Die Buttons sind dynamisch und verändern sich je nach möglichen Befehlen. Durch klicken auf das weiße Fragezeichen (in der grauen Leiste) können Hilfetexte angezeigt werden, diese erklären die einzelnen Buttons.

Hinter dem in Settings ausgewählten Prozessor steht entweder (Atmel, AVR) oder (ARM). Dies zeigt an welcher Programmierer verwendet wird. Für AVR (Abbildung 14) wird die 10 auf 6 polige Adapterplatine genutzt für ARM (Abbildung 15) die 10 auf 20 polige Adapterplatine.

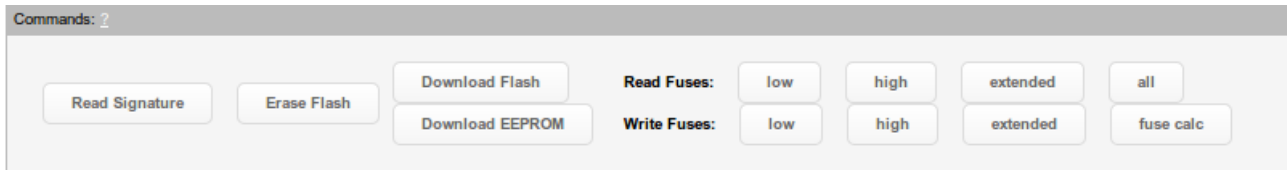


Abbildung 14: Buttons AVR

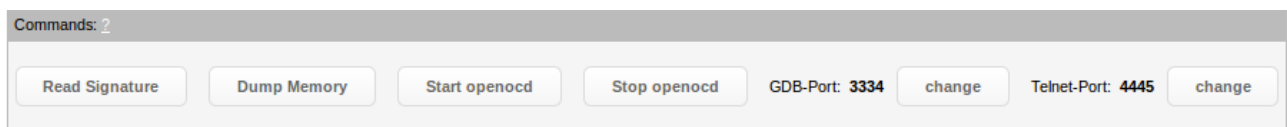


Abbildung 15: Buttons ARM

5.1.4. Programmieren einer Firmware

Um eine neue Firmware hochzuladen, klickt man unter „Upload Firmware“ den Durchsuchen Button. Anschließend kann man ein Hexfile auf seinem lokalen PC auswählen. Es wird automatisch als „Temp File“ angezeigt, da es nach dem Abstecken wieder verloren gehen würde. Um eine Firmware zu einem späteren Zeitpunkt und/oder an einem anderen PC wiederverwenden zu können ist es notwendig die Firmware im Flash Archive zu speichern. Dies geschieht über den „safe profile“-Button.

Wichtig: Vor Hochladen des Tempfiles müssen die dafür benötigten Settings (Abbildung 13) ausgewählt werden, da diese für das Programmieren genutzt werden. Soll eine EEPROM Datei hochgeladen werden, muss die entsprechende Checkbox aktiviert werden.

Sobald ein Tempfile hochgeladen wurde (Abbildung 16), kann dieses auf das Targetboard programmiert werden. Zusätzlich besteht die Möglichkeit dieses wieder vom USBprog 5.0 herunterzuladen oder zu löschen. Durch Anklicken der "Autoflash"-Checkbox wird das Tempfile direkt nach dem Upload auf das Board geflashed.

Der "safe Profile"-Button dient dazu das Tempfile, mit den für das Tempfile gespeicherten Settings, in das Flash Archive zu speichern. Nach dem Klicken des Buttons wird man aufgefordert eine Beschreibung einzugeben unter der das Profil gespeichert wird. Es ist zu empfehlen, falls man verschiedene Boards mit gleichem Prozessor hat, die jeweils eingestellte Spannung in die "Description" einzufügen.

Upload Firmware: ?


Description	Firmware	Processor	Size	Action	Flash
Temp File	yourpgm.hex	m16	0.676kb	 	 program

Keine Datei ausgewählt.

☐ Autoflash after upload

Abbildung 16: Temporäre Firmware in der ARM Oberfläche

Upload Firmware: ?

Description*	Firmware	Processor	Size	Action	Flash
Temp File	myprog(4).hex	m16	0.672kb	 	 program

Keine Datei ausgewählt.

☐ Autoflash after upload
☐ EEPROM

Abbildung 17: Temporäre Firmware in der AVR Oberfläche

5.1.5. Nutzung des Flash Archivs

Die Dateien Im Flasharchive (Abbildung 18) verhalten sich wie das Tempfile ("X" zum löschen, "Pfeil nach unten" zum Downloaden und "program" zum Programmieren des Targets). Durch Klicken auf die jeweilige Description kann diese jederzeit verändert werden.

Flash Archive: ?




Description	Firmware	Processor	Size	Action	Flash
1	yourpgm.hex	ATMEGA16	0.676kb	 	 program

Abbildung 18: Flasharchive

5.1.6. Einspielen eines Updates für USBprog 5.0

Siehe Punkt 5: Firmware Update und Punkt 3.2 Bedienung per Konsole

5.1.7. Kommandozeilen Befehle

Hier (Abbildung 19,20) stehen Beispielfehle, zum Benutzen mit dem Kommandozeilentool, für den ausgewählten Prozessortypen. Nur "your Processor" muss mit dem gewünschten Prozessor ausgetauscht werden. Zum Beispiel m16 für Atmega16.

Command Lines:	
Read Signature	python embeddedprog.py --processor "your Processor" --speed 2
Flash	python embeddedprog.py --processor "your Processor" --flash-write /tmp/yourprog.hex
Start openocd	python embeddedprog.py --processor "your Processor" --gdb "start"
Stop openocd	python embeddedprog.py --processor "your Processor" --gdb "stop"
Dump Memory	python embeddedprog.py --processor "your Processor" --dump "mdw addr count"

Abbildung 19: Kommandozeilenbefehle ARM

Command Lines:	
Read Signature	python embeddedprog.py --processor "your Processor" --speed 2
Flash write	python embeddedprog.py --processor "your Processor" --flash-write yourprog.hex
Flash read	python embeddedprog.py --processor "your Processor" --flash-read yourprog.hex
EEPROM write	python embeddedprog.py --processor "your Processor" --eeprom-write yourprog.hex
EEPROM read	python embeddedprog.py --processor "your Processor" --eeprom-read yourprog.hex
Erase	python embeddedprog.py --processor "your Processor" --delete
Read Fuse High	python embeddedprog.py --processor "your Processor" --fuse-read-high
Read Fuse Low	python embeddedprog.py --processor "your Processor" --fuse-read-low
Read Fuse Extended	python embeddedprog.py --processor "your Processor" --fuse-read-extended
Write Fuse High	python embeddedprog.py --processor "your Processor" --fuse-write-high "fusebits"
Write Fuse Low	python embeddedprog.py --processor "your Processor" --fuse-write-low "fusebits"
Write Fuse Extended	python embeddedprog.py --processor "your Processor" --fuse-write-extended "fusebits"

Abbildung 20: Kommandozeilenbefehle AVR

5.2. Bedienung Per Konsole

Die Bedienung per Konsole ist anspruchsvoller als die Benutzung des Browsers aber sie bietet auch mehr Möglichkeiten sowie mehr Infos über den Vorgang des Programmierens.

5.2.1. Öffnen der Konsole

Linux:

drücken sie strg + alt + t.

Windows:

drücken sie die windows taste + r und geben in ausführen (abbildung 21) cmd ein.

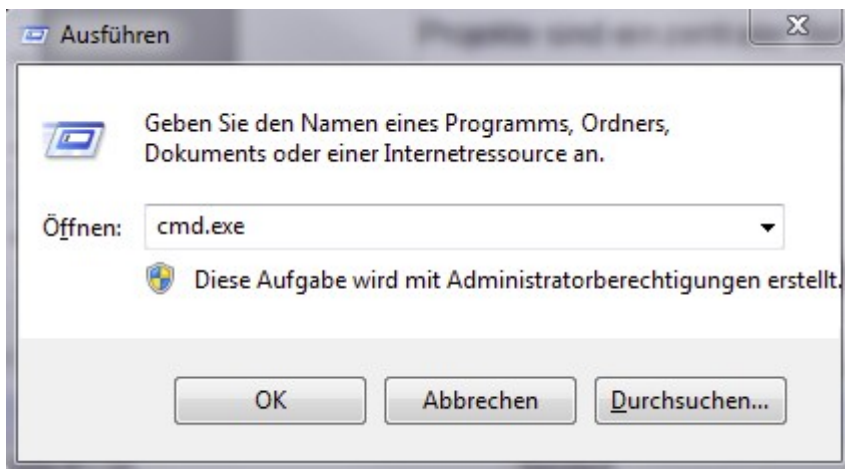


Abbildung 21: Ausführen mit Windows

Mac:

drücken sie cmd + Leertaste und geben in Spotlight (Abbildung 22) Terminal ein.



Abbildung 22: Spotlight zum Aufruf des Terminal in MacOS

5.2.2. Download des Kommandozeilentools

Das Kommandozeilentool `embeddedprog.py` kann entweder vom internen browser (siehe abbildung 23) oder direkt aus dem GIT heruntergeladen werden.

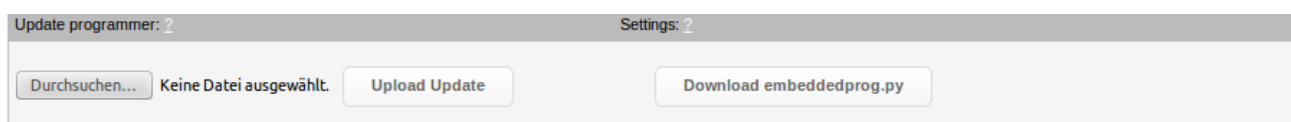


Abbildung 23: update programmer/ download Konsolen tool

5.2.3. Verwendung des Kommandozeilentools

Mit der Option „help“ kann man sich alle weiteren verfügbaren Unteroptinen anzeigen lassen.

Linux/Mac:

```
python embeddedprog.py --help
```

oder

```
python kompletter_pfad_zum_Konsolentool/embeddedprog.py --help
```

Windows:

```
embeddedprog --help
```

5.2.4. Übersicht der Befehle

Alle hier nicht angegebenen Befehle (durch --help anzeigbar) sind für den Browser oder andere Programme gedacht. Es ist davon abzuraten diese ohne Tiefes Verständniss des Programms zu benutzen.

Befehle	Beschreibung
- v	Gibt einen "verbose output", gibt nur infos über das server client Programm des USBprogs und verändert nicht die ausgabe des openocd/avrdudes
--processor	Erwartet den zu programmierenden Prozessor. Muss immer angegeben werden. Falls nur der Processor angegeben wird und ansonsten kein Befehl wird automatisch die Signatur gelesen.
--speed	Momentan nur für AVR processoren. Gibt die Geschwindigkeit an mit der gelesen/geschrieben wird. Default ist 2. 1=fast, 2=normal, 3=slow
--show-all	Zeigt sämtliche unterstützte Processoren an
--signature	Liest die Signatur des Prozessors
--voltage	Hier kann die Spannung mit der programmiert werden soll angegeben werden. 1 für 1V8, 3 für 3V3, 5 für 5V. Falls nichts angegeben wird, wird die zuletzt verwendete Spannung genutzt

Tabelle 1: Allgemeine Kommandozeilentool Befehle

Befehle	Beschreibung
--eeprog-ip	Hier soll die IP des USBprog 5.0 angegeben werden. (Standard 10.0.0.1). Falls nichts angegeben ist wird die Standard Adresse genutzt.
--eeprog-port	Hier soll der Port des USBprog 5.0 angegeben werden. (Standard 8888). falls nichts angegeben ist wird der standart Port genutzt.

Tabelle 2: Kommandozeilentool Config

Befehle	Beschreibung
--flash-write	Schreibt ein Programm auf das Target Board. Erwartet absolute oder relative Pfadangaben.
--dump	<p>Dump memory mit dem Openocd. Erwartet den Befehl in folgender Form:</p> <p>mdw [phys] addr [count]</p> <p>mdw = 32-bit word mdh = 16-bit halfword mdb = 8-bit byte</p> <p>When the current target has an MMU which is present and active, addr is interpreted as a virtual address. Otherwise, or if the optional phys flag is specified, addr is interpreted as a physical address. If count is specified, displays that many unit</p>
--gdb	Dient dazu Openocd zu starten/stoppen damit eine gdb verbindung aufgebaut werden kann. Erwartet den Befehl 'start' oder 'stop'

Tabelle 3: Kommandozeilentool ARM Befehle

Befehle	Beschreibung
--flash-read	Liest den Flash des Targetboards. Erwartet den Dateinamen der zu speichernden Datei.
--flash-write	Schreibt ein Programm auf das Target Board.
--eeprom-read	Liest den EEPROM des Target Boards. Erwartet den Dateinamen der zu speichernden Datei.
--eeprom-write	Schreibt den Inhalt einer Datei in den EEPROM des Target Boards.
--fuse-read-low	Liest die LOW Fusebits aus.
--fuse-read-high	Liest die HIGH Fusebits aus.
--fuse-read-extended	Liest die EXTENDED Fusebits aus.
--fuse-write-low	Setzt die LOW Fusebits. Erwartet die Form 0xFF
--fuse-write-high	Setzt die HIGH Fusebits. Erwartet die Form 0xFF
--fuse-write-extended	Setzt die EXTENDED Fusebits. Erwartet die Form 0xFF
--lockbits-read	Liest die lockbits aus.
--lockbits-write	Setzt die lockbits. Erwartet die Form 0xFF
--lockbits-write-erase	Setzt die lockbits und löscht das device. Erwartet die Form 0xFF
--delete	Löscht den Flash des Target Boards.
--erase	Löscht den Flash des Target Boards.

Tabelle 4: Kommandozeilentool AVR Befehle

6. Einrichten des USBprog 5.0 für Atmel Studio

Falls die Setup.exe noch nicht installiert wurde muss dies jetzt geschehen. Die aktuell Version kann man sich über die interne Weboberfläche herunterladen. Die Oberfläche ist über die Adresse <http://10.0.0.1> erreichbar.

6.1. Einrichten der Funktionen

In der Menüleiste findet man in dem Reiter "Extras" den Punkt "Externe Tools...". (siehe Abbildung 24)

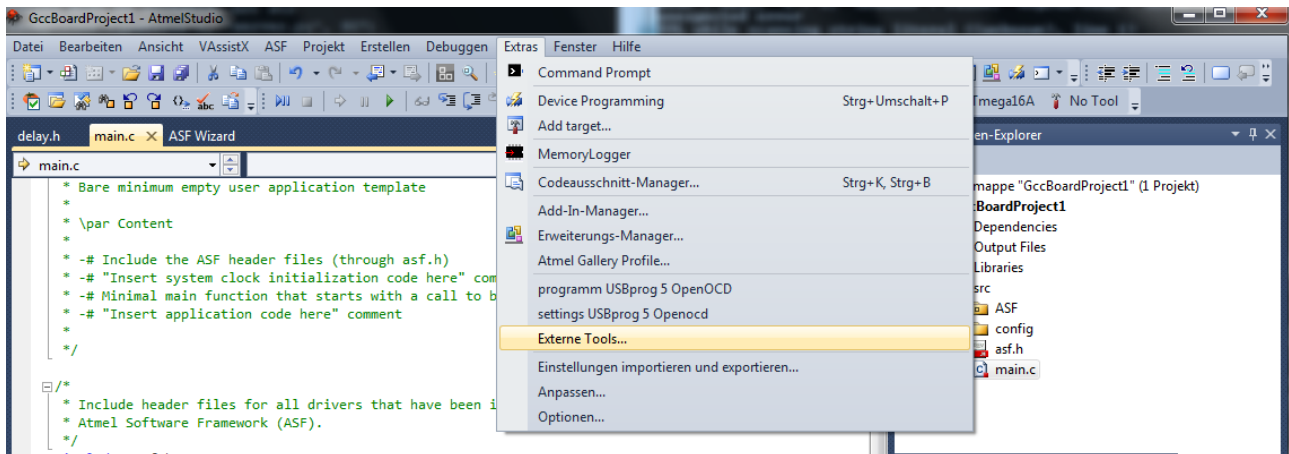


Abbildung 24: Externe Tools

6.1.1. Menüeintrag „Program“

Um den Menüeintrag „Program“ im Atmel Studio zu erhalten ist folgendes „Externe Tool“ zu erstellen. (siehe Abbildung 25)

- Titel: Program with USBprog 5.0 Openocd
- Befehl: embeddedprog.exe
- Argumente: --atmel-studio "\${TargetPath}"

Der Ausdruck `${TargetPath}` ist der komplette Pfad zur erstellten ".elf" file des Projekts. Man kann sie einfügen, durch Klick auf den Pfeil neben dem Feld "Argumente" und dort Zielpfad auswählt.

- Das Feld "Ausgangsverzeichnis" kann leer gelassen werden.

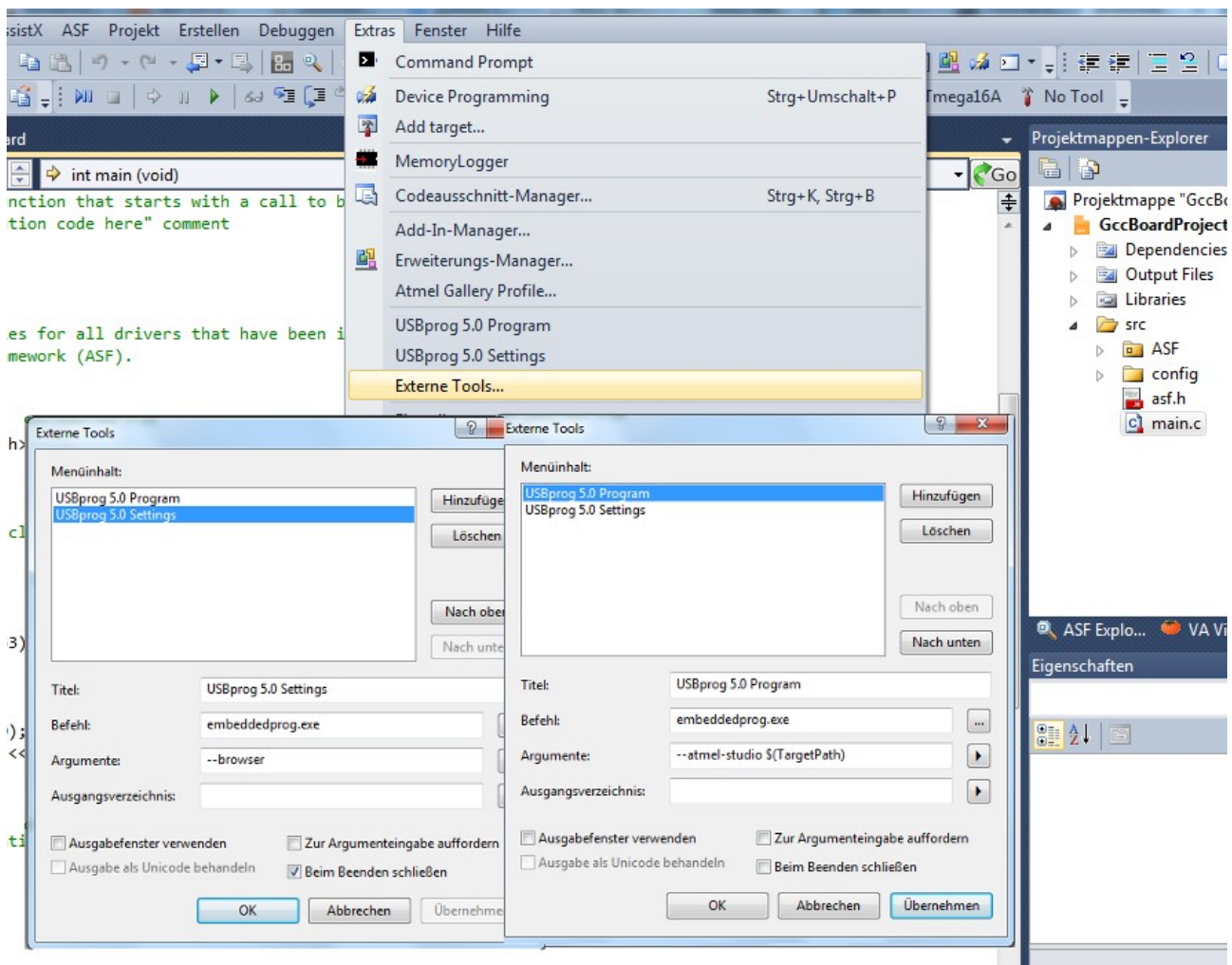


Abbildung 25: Atmel-Studio Funktionen

6.1.2. Menüeintrag „Open Settings“

Um vom Atmel Studio aus die Einstellungen des USBprog 5.0 zu ändern, legen wir einen weiteren Menüeintrag an. Wir nennen diesen „Open Settings“ (siehe Abbildung 25).

- Titel: Open Settings in Browser USBprog 5.0
- Befehl: embeddedprog.exe
- Argumente: --browser
- Das Feld "Ausgangsverzeichnis" kann leer gelassen werden.

6.2. Benutzung der Menüeinträge

Wenn die beiden Menüpunkte eingerichtet sind und alle Tools entsprechend installiert, können Sie den USBprog 5.0 benutzen. Erstellen Sie Ihr Projekt einfach normal und programmieren Sie über das externe Tool (siehe Abbildung 26).

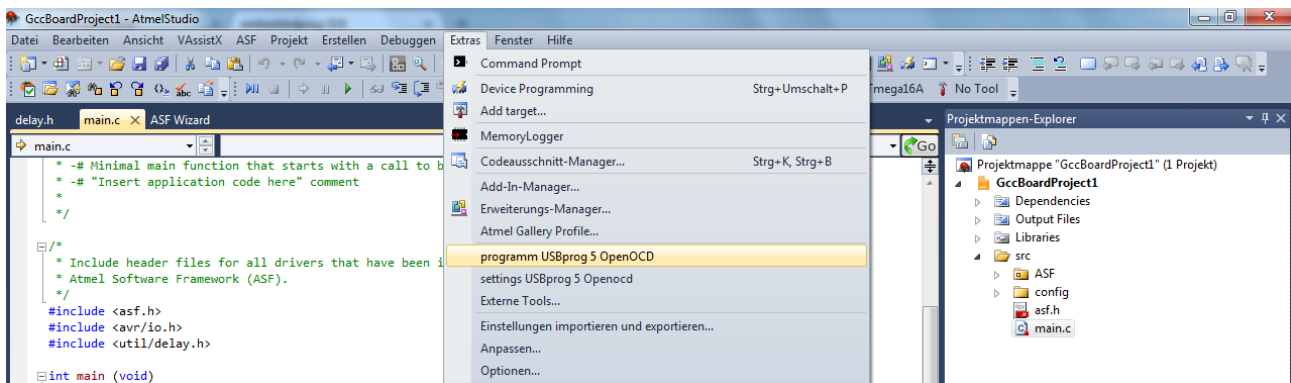


Abbildung 26: Externes Tool Starten

Falls Sie in dem Browser noch keinen Prozessor gewählt haben, wird dieser automatisch aufgerufen und Sie werden aufgefordert den Prozessor, die Geschwindigkeit sowie die Spannung auszuwählen. Über die "embeddedprog.exe" wird das externe Tool die dort eingestellten Werte übernehmen. (siehe Abbildung 27).

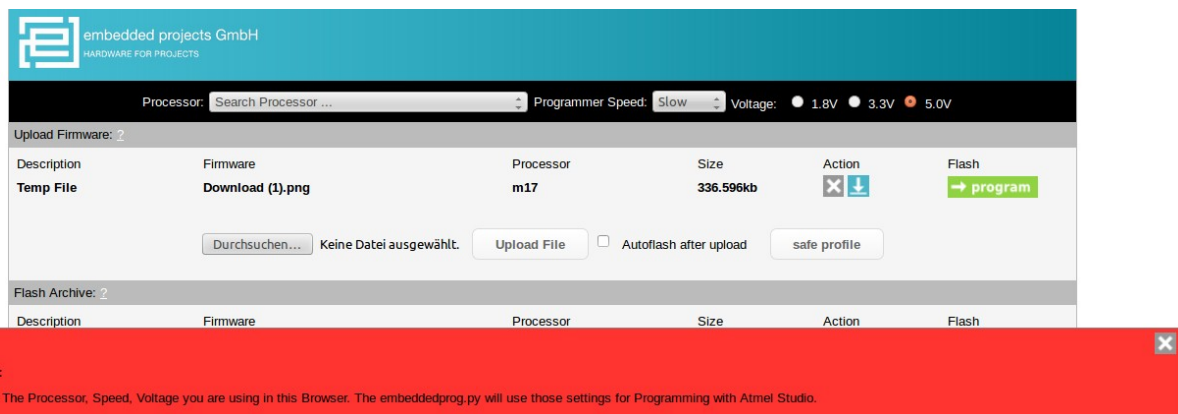


Abbildung 27: Select Settings (Wichtig ist die rote Meldung)

Sobald die Werte korrekt ausgewählt sind wird die kompilierte Datei bei einem Klick auf das externe Programm Tool geflashed. (siehe Abbildung 28).

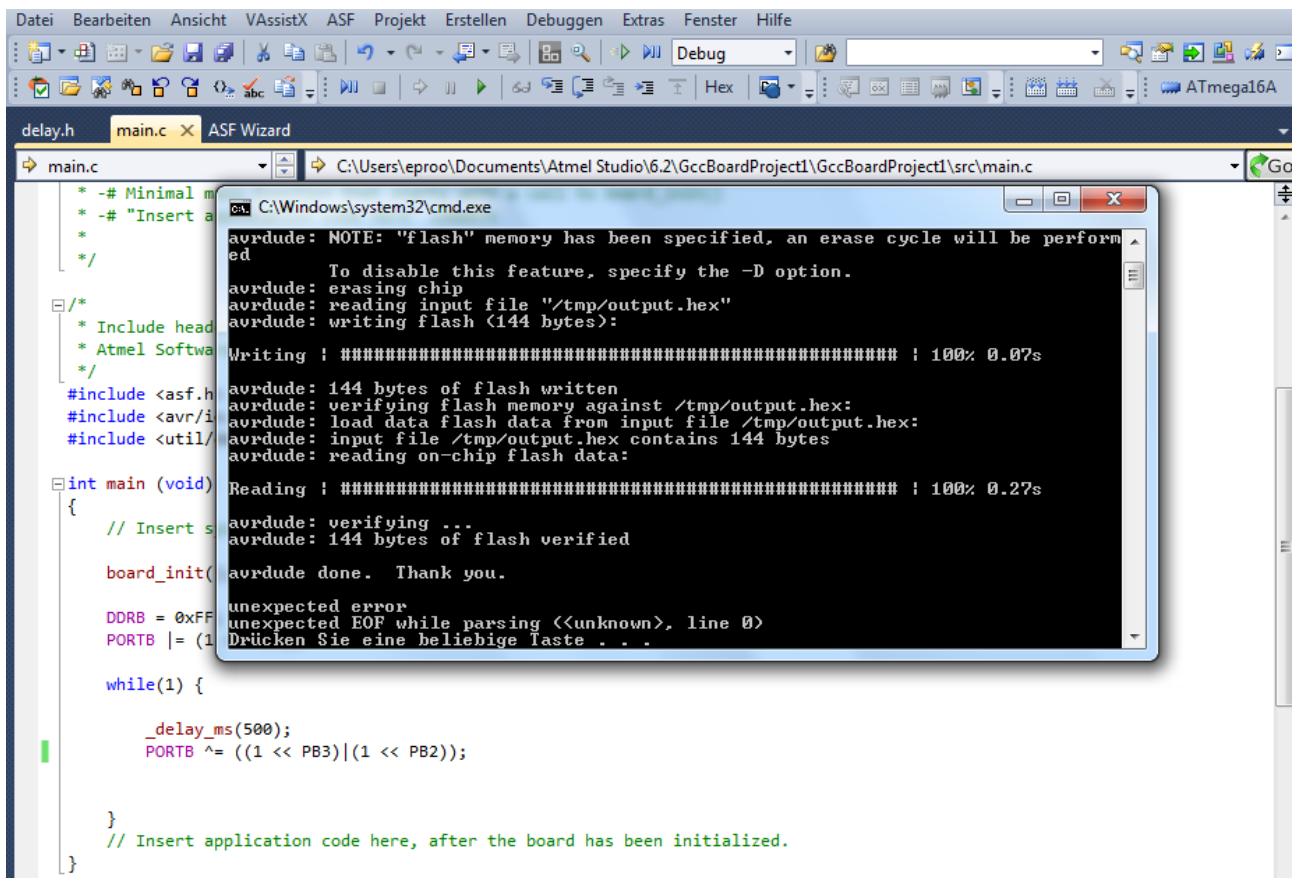


Abbildung 28: Atmel Studio nach Flash

Sie können die Settings jederzeit im Browser verändern. Benutzen Sie dazu das Tool "Open Settings in Browser" oder rufen Sie im Browser selbst folgende Adresse auf: 5975;95;118;97;86

5976;9 <http://10.0.0.1>

7. Einrichten des USBprog 5.0 als Programmierer für Eclipse

Der USBprog kann relativ simpel in Eclipse eingebaut werden. Hierzu muss zuerst das Fenster External Tools Configuration aufgerufen werden und dort dann die Location des Kommandozeilen-Tools, sowie die entsprechenden Argumente angegeben werden (siehe Abbildung 29).

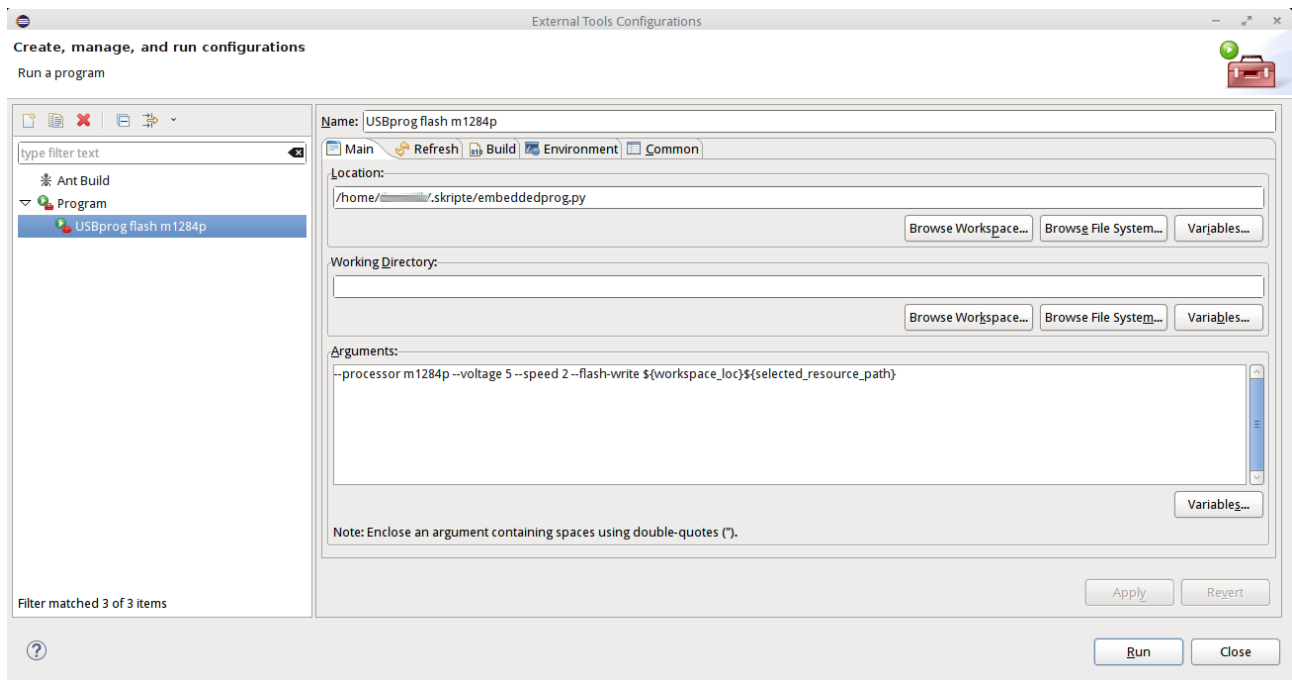


Abbildung 29: External Tools Eclipse

Wichtig ist hierbei zu beachten dass, die Feldbefehle von Eclipse richtig unter Argumente eingegeben werden (siehe Abbildung 30).

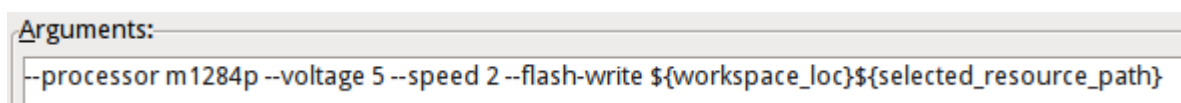


Abbildung 30: Eclipse Arguments

Unter dem Tab Build sollte „Build before launch“ und „The project containing the selected resource“ ausgewählt sein (siehe Abbildung 31).

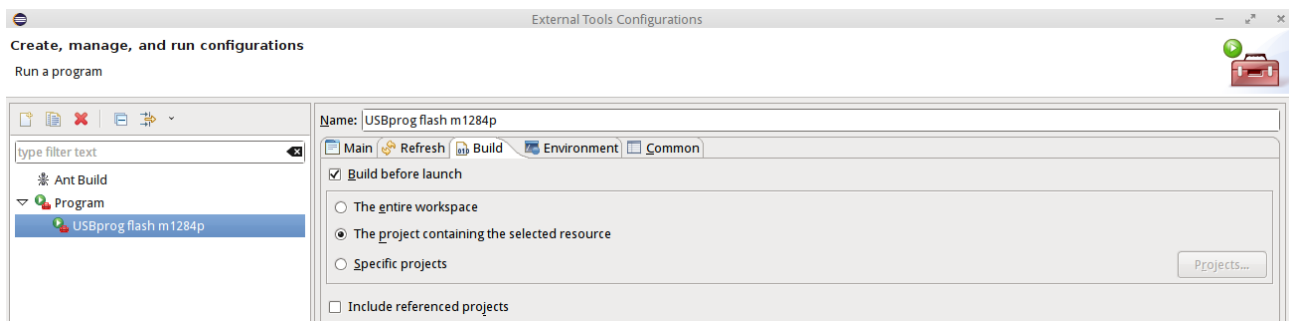


Abbildung 31: BuildTab Eclipse

Als Letztes muss nur noch der Tab Common angepasst werden. Dort sollte „Local file“, „External Tools“, „Default – inherited (UTF-8)“, „Allocate console“ sowie „Launch in Background“ ausgewählt sein (siehe Abbildung 32)

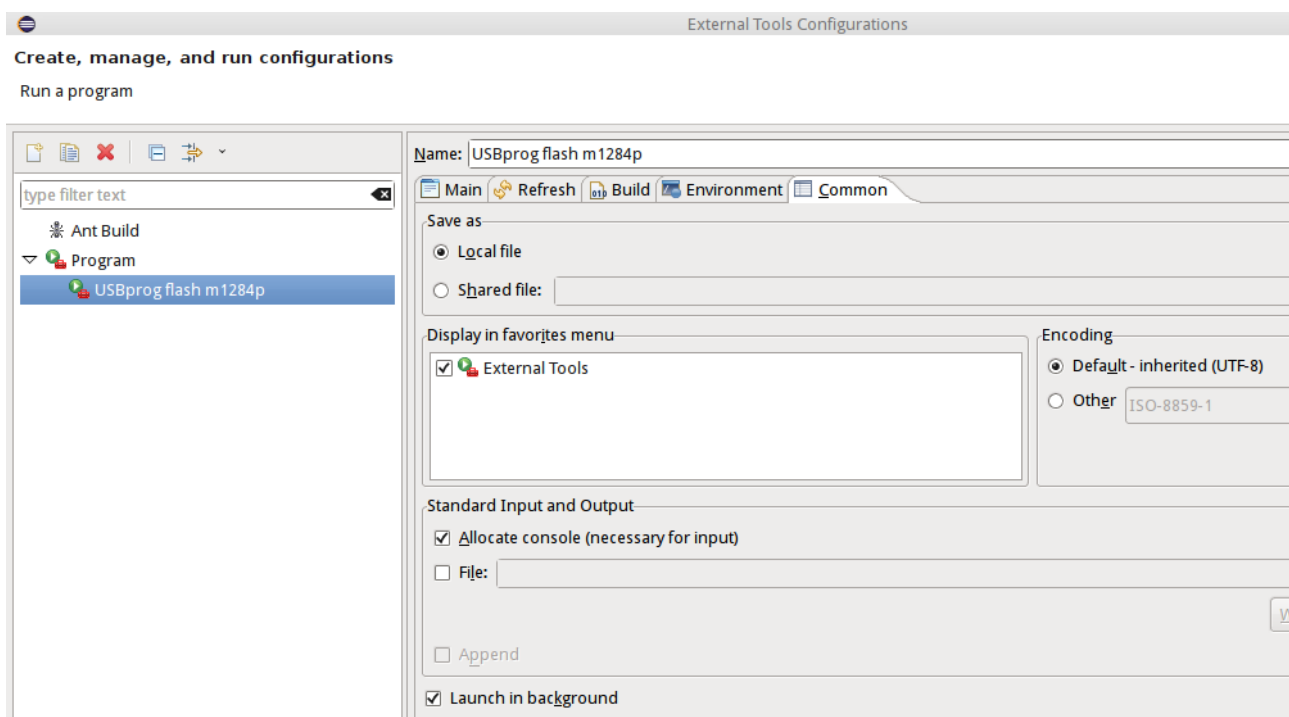


Abbildung 32: Common Tab Eclipse

Danach kann der USBprog 5.0 in Eclipse Verwendet werden(siehe Abbildung 33).

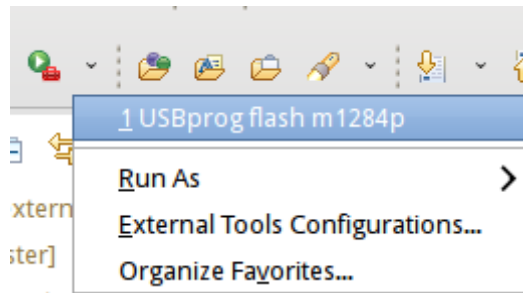


Abbildung 33: Flash with Eclipse

8. Update Firmware

Die aktuelle Firmware ist stets unter folgenden Link erhältlich:

<https://github.com/embeddedprojects/usbprog5/raw/master/update/update.tar.gz>

Um dieses Update einzuspielen, müssen sie zuerst die Datei update.tar.gz herunterladen.

Anschließend müssen Sie auf die Website des USBprog 5.0 gehen - diese ist standardmäßig unter <http://10.0.0.1> zu erreichen.

Dort klicken sie auf Durchsuchen unter dem Punkt Update Programmer (siehe Abbildung 34).

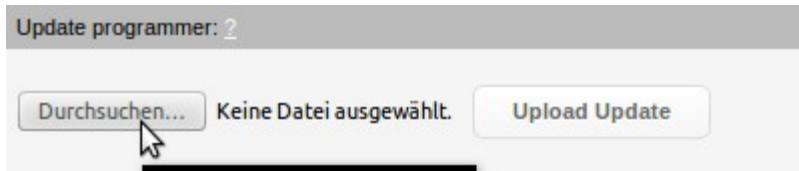


Abbildung 34: Update Firmware

Suchen sie nun die „update.tar.gz“ in ihrem Dateimanager und klicken sie auf öffnen.

Klicken sie nun auf „Upload Update“ siehe Abbildung 35.



Abbildung 35: Upload Firmware Update

Nun erscheint ein Fenster, welches „update in progress“ anzeigt, nach ein paar Sekunden wechseln sie automatisch in die update.php, welche wiederum (diesmal in größerer Schrift) „update in progress“ anzeigt. Sobald das Update eingespielt ist werden sie wieder auf die Standard USBprog Seite

weitergeleitet.

9. Debuggen mit OpenOCD

Mit dem USBprog können ARM Prozessoren mittels JTAG Schnittstelle gedebuggt werden. Entweder kann man das über die Kommandozeile oder mittels Eclipse durchführen.

9.1. Debuggen mit Kommandozeile

9.1.1. Linux

Um mit Linux über die Konsole zu debuggen, muss zuerst ein entsprechendes gdb Programm installiert sein.

Im Falle eines stm32f1x kann beispielsweise, gcc-arm-none-eabi-gdb (<https://launchpad.net/gcc-arm-embedded>) als gdb Client verwendet werden.

Zum Starten des Programms muss zuerst der gdb Server auf dem USBprog5.0 laufen. Dieser kann entweder über die embeddedprog.py/exe gestartet werden, mit dem Befehl:

```
python embeddedprog.py --processor "processor" --gdb start
```

oder über die Webseite des USBprog 5.0 (<http://10.0.0.1>) mit dem entsprechenden Button.

Befehl	Beschreibung	Ausgabe (bei einem stm32f1x)
gcc-arm-none-eabi-gdb <File>	Startet den gdb Client auf dem PC, falls ein File mit angegeben ist kann dieses in anderen Befehlen weiter verwendet werden	GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20131129-cvs
target remote ip:port (10.0.0.1:3333)	Verbindet den gdb Client mit dem USBprog5.0 (In Kommandozeile des GDBs eingeben)	(gdb) target remote 10.0.0.1:3333 Remote debugging using 10.0.0.1:3333
monitor	Wird benutzt, um mit einem Remote-Target zu kommunizieren	
monitor reset	Resettet ein Remote-Target	(gdb) monitor reset JTAG tap: stm32f1x.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x3) JTAG tap: stm32f1x.bs tap/device found: 0x06420041 (mfg: 0x020, part: 0x6420, ver: 0x0)
monitor halt	Hält das laufende Programm an.	(gdb) monitor halt

		target state: halted target halted due to debug-request, current mode: Thread xPSR: 0x61000000 pc: 0x08000074 msp: 0x20001ff0
monitor step	Geht in dem laufenden Programm einen Schritt (step) weiter	(gdb) monitor step target state: halted target halted due to single-step, current mode: Thread xPSR: 0x61000000 pc: 0x08000074 msp: 0x20001ff0
list	Zeigt den aktuellen Code an, falls ein File geladen wurde	(gdb) list 23 24 25 for(i=0; i < 10; i++){ 26 int_debug = int_debug + i * (i*i) + int_debug_value; 27 }
load	Lädt das beim Starten mitgegebene File auf den Prozessor	(gdb) load Loading section vectors, size 0x40 lma 0x8000000 Loading section .text, size 0x90 lma 0x8000040 Loading section .data, size 0x4 lma 0x80000d0 Start address 0x8000044, load size 212
q,quit	Beendet die Debug-Session	Detaching from program: main.elf, Remote target Ending remote debugging.

Beispielhafter Ablauf des Debuggens. Auf PC Seite ruft man den ARM GDB mit der entsprechenden .elf Datei auf. Mittels -g kann man beim Compilieren mit dem GCC noch angeben, dass Debuginformationen mit in das .elf integriert werden.

```
:~$ /gcc-arm-none-eabi/bin/arm-none-eabi-gdb /Pfad zur Datei/main.elf
```

Anschliessend erhält man folgende Ausgabe:

```
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20131129-cvs
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-linux-gnu --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
```

Reading symbols from /home/Pfad zur Datei/main.elf...done.

Jetzt kann man sich mit dem USBprog über das Netzwerk verbinden:

```
(gdb) target remote 10.0.0.1:3333
```

```
main () at main.c:36
36      i=1 ;
```

Den Prozessor anhalten:

```
(gdb) monitor reset halt
```

```
JTAG tap: stm32f1x.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part: 0xba00,
ver: 0x3)
JTAG tap: stm32f1x.bs tap/device found: 0x06420041 (mfg: 0x020, part: 0x6420,
ver: 0x0)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000044 msp: 0x20002000
```

Die Firmware laden:

```
(gdb) load
```

```
Loading section vectors, size 0x40 lma 0x8000000
Loading section .text, size 0x90 lma 0x8000040
Loading section .data, size 0x4 lma 0x80000d0
Start address 0x8000044, load size 212
```

```
(gdb) monitor reset halt
```

```
JTAG tap: stm32f1x.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part: 0xba00,
ver: 0x3)
JTAG tap: stm32f1x.bs tap/device found: 0x06420041 (mfg: 0x020, part: 0x6420,
ver: 0x0)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000044 msp: 0x20002000
```

Das Programm durchlaufen:

```
(gdb) step
```

```
RESET_Handler () at startupcode.c:8
8      main();
```

```
(gdb) step
```

Note: automatically using hardware breakpoints for read-only addresses.

```
main () at main.c:18
18      int_debug = int_debug + int_debug_value;

(gdb) step

26      int_debug = int_debug + i * (i*i) + int_debug_value;

(gdb) quit

A debugging session is active.

    Inferior 1 [Remote target] will be detached.

Quit anyway? (y or n) y
```

9.2. Debuggen mit Eclipse

- Eine Anleitung folgt.

10. USBprog 5.0 Systemdetails

10.1. Open-Source Programme

Auf dem USBprog 5.0 arbeitet im Hintergrund ein Linux System. Die Anbindung der Targets erfolgt über GPIO und z.T. über die SPI Schnittstelle des Prozessors um schnelle JTAG Sequenzen erzeugen zu können.

Der Kern der Anwendung auf dem USBprog 5.0 ist das Programm server.py. Dieses bietet die API für den Browser und das Kommandozeilentool per Netzwerk an.

Die Webseite, welche in PHP geschrieben ist, liegt im Ordner /var/www. Für die Programmierung und das Debuggen der vielen Prozessoren wurden die Standardanwendungen Avrdude und OpenOCD verwendet bzw. entsprechend angepasst. Alle Quelltexte zu USBprog 5.0 findet man im GIT des USBprog 5.0.

10.2. GPIO Belegung

OpenOCD	Avrdude	gpios	pin	pin	gpios	avrdude	OpenOCD
TDO	MISO	GPIO 18	1	2	vcc		
TCK		GPIO 17	3	4	GPIO 16		
TDI	RESET	GPIO 2	5	6	GPIO 13		TMS
	SCK	GPIO 20	7	8	GPIO 12		TRST
	MOSI	GPIO 19	9	10	gnd		

Table 5: GPIO 10-polige Buchse

Beschreibung	Pin	Pin	Beschreibung
GPA0	1	2	GPA1
GPA2	3	4	PWM
SCL	5	6	SDA
MOSI	7	8	MISO
SYSCLOCK	9	10	SPI SCK
GPIO14	11	12	GPIO11
3.3V	13	14	GND

Table 6: 14-polige GnuBlin/EEC Buchse

11. API

Die gesamte Kommunikation mit dem USBprog 5 läuft über Sockets ab. Es wird JSON verwendet um Befehle zu übertragen. Anfragen werden an die IP Adresse des USBprog und Port 8888 gesendet.

Die API wird aktuell überarbeitet. Falls Sie diese benötigen melden Sie sich bitte bei uns.

11.1. JSON Befehle

JSON	Typ	Erwartete Werte	Beschreibung
"v":	int	1-3	Gibt an wie detailliert die Ausgabe ist. V=3 ist nur zum debuggen des usbprog 5 gedacht.
"processor":	String	Ein processor	Wählt aus welcher programmer (avrdude/openocd) verwendet wird und welcher processor zu programmieren ist.
"signature":	bool	True False	Gibt die signatur des gewählten processors aus. Ist nicht benötigt da die signatur auch ausgegeben wird wenn nur der processor angegeben wird

11.2. JSON Beispiel

11.2.1. Signatur abfragen

Request (wird von Client aus gesendet)

```
{"processor":"lpc2102","command":"readsignature","speed":"2","voltage":"3"}
```

Response (Antwort vom Server)

```
{"result":"gelesenesignatur"}
```

11.2.2. Firmware flashen

Beim Flashen auf diese Art und Weise muss die Firmware zuvor auf dem USBprog in dem Ordner /var/www/save liegen. Die Datei kann per SCP oder über den Upload auf der Weboberfläche übertragen werden.

Request

```
{"processor":"lpc2102","command":"load","web":"true","filename":"test.hex","filecontent":"base64kodierter Datei"}
```

Response (im Erfolgsfall)

```
{"result":"1"}
```


Abbildungsverzeichnis

Abbildung 1: USBprog 5.0.....	5
Abbildung 2: ARM 10 auf 20.....	6
Abbildung 3: AVR 10 auf 6.....	6
Abbildung 4: Systemsteuerung öffnen.....	8
Abbildung 5: Geräte-Manager wählen.....	8
Abbildung 6: RNDIS Gadget aktualisieren.....	9
Abbildung 7: Auf dem Computer nach Software suchen.....	9
Abbildung 8: Aus Liste auswählen.....	10
Abbildung 9: Netzwerkadapter.....	10
Abbildung 10: Microsoft Corporation Remote NDIS.....	11
Abbildung 11: Browser.....	12
Abbildung 12: Konsole.....	12
Abbildung 13: Settings.....	12
Abbildung 14: Buttons AVR.....	13
Abbildung 15: Buttons ARM.....	13
Abbildung 16: Temporäre Firmware in der ARM Oberfläche.....	14
Abbildung 17: Temporäre Firmware in der AVR Oberfläche.....	14
Abbildung 18: Flasharchive.....	14
Abbildung 19: Kommandozeilenbefehle ARM.....	15
Abbildung 20: Kommandozeilenbefehle AVR.....	15
Abbildung 21: Ausführen mit Windows.....	16
Abbildung 22: Spotlight zum Aufruf des Terminal in MacOS.....	16
Abbildung 23: update programmer/ download Konsolen tool.....	16
Abbildung 24: Externe Tools.....	20
Abbildung 25: Atmel-Studio Funktionen.....	21
Abbildung 26: Externes Tool Starten.....	22
Abbildung 27: Select Settings (Wichtig ist die rote Meldung).....	22
Abbildung 28: Atmel Studio nach Flash.....	23
Abbildung 29: Update Firmware.....	24
Abbildung 30: Upload Firmware Update.....	24

Tabellenverzeichnis

Tabelle 1: Allgemeine Kommandozeilentool Befehle.....	17
Tabelle 2: Kommandozeilentool Config.....	18
Tabelle 3: Kommandozeilentool ARM Befehle.....	18
Tabelle 4: Kommandozeilentool AVR Befehle.....	19
Table 5: GPIO 10-polige Buchse.....	28
Table 6: 14-polige GnuBlin/EEC Buchse.....	29

Stichwortverzeichnis